VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETECTION OF NETWORK ATTACKS BASED ON NETFLOW DATA

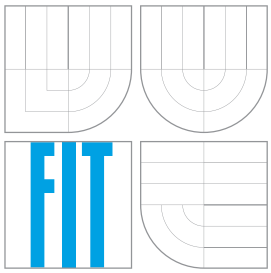
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VOJTĚCH KULIČKA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE SÍŤOVÝCH ÚTOKŮ NA ZÁKLADĚ NETFLOW DAT

DETECTION OF NETWORK ATTACKS BASED ON

NETFLOW DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VOJTĚCH KULIČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ŽÁDNÍK
SUPERVISOR

BRNO 2009

Abstrakt

V současné době stále pokračuje dlouhodobý trend nárůstu kyberkriminality takřka po celém světě. Tato práce se zabývá stále sílící problematikou bezpečnosti síťového provozu, konkrétně detekcí útoků. V rámci práce je navržen program pro detekci anomálií na síti na základě NetFlow dat, za účelem důkladnější ochrany běžných uživatelů. Program je realizován metodou TCM-KNN využívající statistických odlišností útoků, čímž umožňuje zaznamenat i jejich nové, dříve neviděné instance

Abstract

With rising popularity of the internet there is also rising number of people misusing it. This thesis analyzes the problem of network attack detection based on NetFlow data. A program is designed to point out anomalous behaviour by analyzing the flow records using data mining techniques. The method of TCM-KNN utilizing the fact that attacks statistically deviate is implemented. Thus even new types of attacks are detected

Klíčová slova

IDS, detekce anomalií, síťová bezpečnost

Keywords

IDS, anomaly detection, network security

Citace

Vojtěch Kulička: Detection of Network Attacks Based on
NetFlow Data, bakalářská práce, Brno, FIT VUT v Brně, 2009

Detection of Network Attacks Based on NetFlow Data

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing.Martina Žádníka

.....
Vojtěch Kulička
May 20, 2009

Poděkování

Chtěl bych především poděkovat Ing.Matrinu Žádníkovi za odbornou pomoc a rodině za podporu.

Contents

Chapter 1

Introduction

I get up in the morning, turn my laptop on, it automatically connects to the Internet and instead of a morning newspaper I read the news using rss from my favorite news sites. I receive only the types of news I am interested in, free of charge and of course up-to-date. Youtube comes next with it's music videos to lighten up my morning, while I am checking my mail and my facebook account.

Popularity of the Internet has grown extremely over the past decade and so have the number of diverse services it accommodates. Google and Microsoft let you to post your family pictures and share them with whoever you want to. TV channels allow you to watch TV shows whenever you wish. Online encyclopedias contain almost any information from any field. Newspapers came online changing a meaning of the words up-to-date. Single player games are losing popularity to massive multiplayer online(MMO) games, where you compete or cooperate with players all over the world. Communicating with people on the other side of the globe has never been so easy and cheap. A headset, a webcam, an internet connection(with proper bandwidth) and a proper software is all it takes to talk and even see anyone anywhere on the planet. There are many other great services naming which would take half of this thesis.

The growth of the Internet has steered the course of most if not all branches of business it's direction. Banks have deployed online banking making paying the bills a little more pleasant activity. Electing officials in advanced countries take place over the internet. A new branch of business was born. Branch to which firms like Yahoo and Google belong to. These firms make money by placing advertisements on their websites, which are daily visited by millions of people. Advertisement is moving from TV-sets to our computers. And of course shopping has changed a lot in past few years. The so called e-shop have experienced a boom. Many people have replaced their stores with e-shops and helpful retailer with reviews from people who have already bought whatever it is you are shopping for.

As we can see the Internet has become a part of our society. It has become a place where people socialize, shop, bet, get information, enjoy themselves and most importantly where they make money. Unfortunately there are people who instead of using the Internet are misusing it or even worse trying to destroy it. Some do it money other for recognition, reputation or just for fun. This thesis tackles the problem and tries to design an algorithm to detect network attacks based on Cisco's NetFlow data.

1.1 NetFlow data

NetFlow is a network protocol developed by Cisco. It is used for collecting records of IP flows on the network. An IP flow is a sequence of connections sharing the same 5-tuple. This 5-tuple consists of protocol, source IP address, source port, destination IP address and finally destination port. A flow record is represented as a set of attributes depending on the version of NetFlow used. NetFlow v5 is the most commonly used and defines a flow with duration of the flow, time and date of flow initiation, TCP flags, number of bytes and packets transferred and naturally the basic 5-tuple. NetFlow is recorded by probes or routers called NetFlow exporters and are sent to NetFlow collector - machine with high storage capacity. The probes do not manipulate with the data streams in any way, just monitor and thus are invisible to a possible attacker. The data are transported from an exporter to a collector via UDP or SCTP(stream control transmission protocol). The network operator then examines the data to see if there are any bottlenecks on their network. It serves to identify major sources of traffic and is priceless when expanding the network. Least but not last use of NetFlow is for security purposes to identify unauthorized access or even attacks.[3][4]

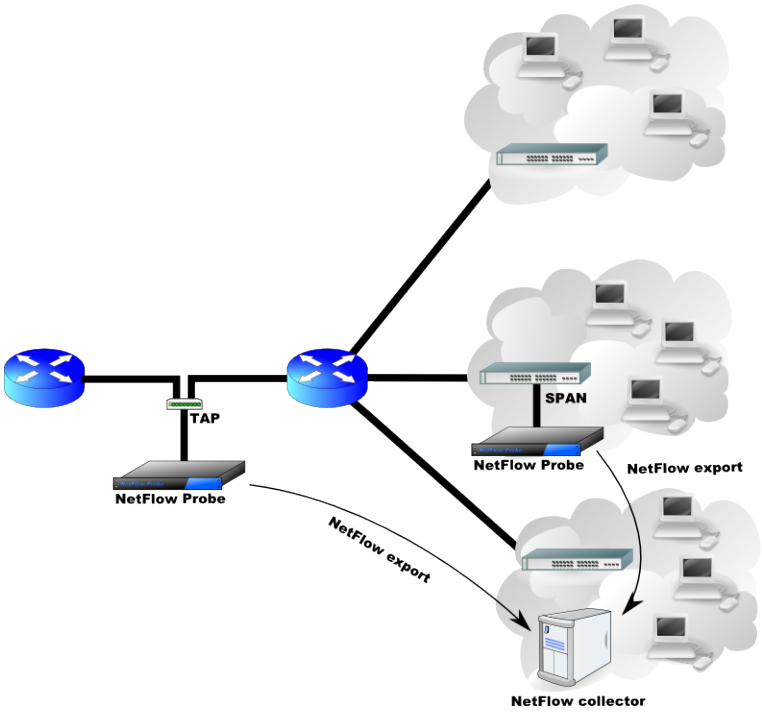


Figure 1.1: NetFlow architecture with probes as NetFlow exporters[3]

Chapter 2

Internet architecture

The Internet was designed to accommodate large number of host and to easily expandable. To better understand the security issues we will go over some of basic principles of the Internet.

2.1 Packet switching

The Internet uses packet switching as opposed to public telecommunication network based on circuit switching. When you wish to make a phone call, you dial a number(address) of a remote machine and a circuit is established. This circuit stays established as long as you don't hang up and no one else can use it, while it is allocated for your call. Even if you are not talking(not transmitting any data) the circuit is still dedicated to you. Packet switching on the other hand does not dedicate the entire route to you, but rather parts of the route as you packets go. Once they have been transmitted over the part of the route other users may use it. This principle ensures full utilization of the line and even load balancing when multiple routes to the desired destination are present.[17]

2.2 Simple core

In order to make the Internet fast, the architects tried to keep the core as simple as possible. All the complexities were moved to end hosts. The backbone routers forward packets based on their IP addresses and do not understand the layers above. They have no way to cheAs new protocols emerge, there is no need to change the core. The protocols are implemented at the end hosts and IP is still used for packet delivery. This way the core stays simple and fast.[14]

2.3 Multipath routing

This principle has made the Internet extremely robust. Whenever a packet is received by a router it looks at it's destination IP address. Then finds proper record in its routing table and sends the packet out a proper interface. If for whatever reason one route fails the routers establish new route via routing protocols and soon continue forwarding packets. Advanced routing protocols allow using different route if the primary one becomes congested to balance load. The core routers do not keep any record of data that passed through it simply because it would require an enormous storage capacity.[17]

Chapter 3

Types of network attacks

The network attacks are of two basic categories. First is using packets with special content to exploit a vulnerability of victim's system. The attacker aims either at getting access to the system and to the data it contains or just making it crash. These attacks are not as common for once the vulnerability is exploited a patch is made to prevent repeated penetration. Attacks of the second category on the other hand use large volumes of data and the attackers' goal is solely to take it down. The victim as a result is unable to serve legitimate users. The second category is called Denial of Service(DoS). DoS has been around for quite some time and it's popularity has been growing exponentially. Number of DoS per year has risen from six in 1988 to 137529 in 2003 according to CERT. These numbers are underestimates for CERT was not informed of all the incidents. CERT has stopped publishing number on DoS attacks in 2004.[17]

3.1 IP Spoofing

IP Spoofing is a technique without which many network attacks would be impossible to perform. It is simply faking source IP address and it is unfortunately very easy to do. You need root/administrator privileges to use raw sockets and you are good to go. Not only it makes it harder(if not impossible) to trace to source of the attack it also gives the attackers more options when choosing type of attack. If you are asking how is it possible the answer is in the key principles of the Internet. The core was designed to be simple and fast and to allow new services to use it without making any changes. Hence it is impossible to implement algorithms for authentication. IP spoofing made it possible to execute attacks such as smurf attack, TCP SYN flood attack or the feared DNS reflection attack, which will be described later on. Although it comes quite handy to the attackers preventing it would not stop DDoS. With large enough botnet there is no need for IP spoofing for the attackers use encrypted messages to command the zombies not to get exposed and discovering several out of tens of thousands of compromised machines does not do the victim much good. [14][18]

3.2 Denial of Service and Distributed Denial of Service

DoS is very easy to execute. All it takes is searching 'DoS howto' and the first result google spits out tells you exactly how to to it. The DoS manual does through a program called xxpoof. xxpoof is an command-line program accepting six rather simple arguments. The

attacker - a script kiddie - does not even have to understand what they mean and for five of them just use default values, which are sufficient. First argument is an IP address of the machine we wish to take out. The other five then specify the attack. The first attack specifying argument is IP address we wish to you instead of our real one. The default value is random, which generates for each burst of data new IP address. Second argument is a set of ports data will be sent to. The default value is also random, which also goes for the rest except for the last. Next arguments are a set of source ports, number of packets to be sent in one burst and finally delay between two bursts. The delay is by default set to 1, which is not that dangerous. The howto of course says to set it to 0 making the data generation continuous and thus even deadlier. The random values for spoofed IP and spoofed port make it very difficult for the victim to filter out the unwanted traffic. Random destination port on the other hand is rather worse than good. If the victim has a properly set up firewall with only few ports open all except for a fraction of the attack traffic might be blocked. This simple program may if running on machine with high enough bandwidth be a source of dangerous DoS attack. This fact is yet another proof of the easiness with which a machine can be brought down.[20]

DDoS is a DoS, but using remotely controlled machines called zombies or agents to fulfill its task. The attackers first need to gain control over a certain amount of zombies called attack network or botnet in order to execute a successful DDoS. The number of zombies needed depends on the type of the attack. Zombie is a poorly secured machine without patches for known exploits. Worms are used to find these machines, break into them, copy their payloads and spread further. They are programmed to patch the vulnerability they used to break into the machine so that it is not taken over by another attacker. The attackers also use emails with malicious attachments, which infect the computers when executed by not knowing users.[14]

DDoS are further divided into simple flooding attacks and sophisticated attacks. The simple flooding attack generates as much data as possible to overwhelm the victim causing a congestion on its network segment and if powerful enough even on an upstream ISP's network. Thousands or even tens and hundreds of thousands zombies are used. Those numbers might seem insanely large, but it has proven very simple to obtain such attack network. To illustrate that getting an army of zombies is very easy let's see couple figures:

- The famous MSBlast worm compromised altogether 9.5 million hosts[10]
- The Sasser worm has infected over 2 million machines according to Microsoft [9]
- The newest worms Downadup or Conflicker exploit a bug in RPC in Microsoft Windows[6]

Prior to launching an attack the attackers first need to get familiar with the victim's network segment and firewall. For these purposes so called network scanners are used. They find open ports on the victim's machine or if there are any compromiseable machines on the victim's network segment. These scanning activities are easily done thanks to free programs like scapy and nmap. Nmap comes natively in many linux distributions and is able to determine what IP addresses are active, what services are running on a particular machine including name and version of the application providing it. Moreover it tells you even what operating systems and what versions is particular machine running, what type of packet filtering, firewalls are in place and much more[**?**]. Scapy has most of the nmap's features and in addition allows you to build your own packets[5].

When launching DDoS attack the attackers do not use all their zombies at once. They start with a portion and periodically check if the site or service is down. If not more

zombies are ordered to attack the victim. Some attackers even enlarge their botnets during the attack. Tools for zombie recruitment are available on underground websites and are easy to use with little knowledge and skill. Amateurs using these toolkits are called script kiddies. Aim of the defenders is to make successful execution of DDoS attack harder so that only the best attackers, which are not so many, prevail.

Programs used to control zombies called bots have greatly improved over the last several years. In the beginnings breaking into remote machines was not even automatic and the bots were able to perform only one type of an attack. Nowadays bots spread by worms, which is a highly automated process. They are updateable and capable of numerous types of attacks and thus much more potent and dangerous. An important bot characteristic is absence of TCP flow control. Whenever the network gets close to being congested the flow control ensures clients sending data to this segment adjust their sending rates. This mechanism unfortunately slows down only sending rates of legitimate users thus actually aiding the attack to make the victim deny service. [14]

3.3 Service and application level attacks

These attacks use vulnerabilities in protocols or applications used by the victim. Whenever an asymmetry exists the attackers gladly take advantage of it. Asymmetry means the attacker uses less resources than the victim. There are many kinds of service and application level attacks, some of which were disabled by proper router setup such as smurf attack but great many are still possible. Here are some of the worst:

3.3.1 TCP SYN attack

This attack is possible due to the unfortunate architecture of TCP. In order to establish a connection one has to complete a three-way handshake. This process is initiated by a client which sends a SYN packet to the server. The server usually using BSD sockets allocates fairly large data structures to serve the client and if everything goes well it replies by sending SYN ACK packet and waits for a ACK packet from the client. As soon as the ACK packet is received, communication starts usually by the client sending a request to the server. When commencing TCP SYN attack the attackers leave the connection half open by not sending a ACK packet to complete the three-way handshake. That might be done either by spoofing client's source IP address and the SYN ACK packet is than delivered to a machine, which is not expecting it and therefore discards it. Or by adjusting the client machine(zombie) to ignore SYN ACK. The problem is that the server may have only so much half open connections and it is very easy to use the capacity up. The allocated structures are freed if the server receives RST packet or when the request times out.[17]

TCP SYN attack is still popular mainly due to small number of zombies needed to successfully commence it.

3.3.2 Distributed reflected denial of service

DRDoS is a special type of DDoS where a public service is misused to amplify the attack. The attacker first either places a large record onto an authoritative DNS server or finds a large enough record already available and then commands the zombies to send queries as in figure 3.1. There are many large records available making it even easier for the attacker to use this type of attack. The zombies are instructed to spoof their IP address

and use victim's address instead. Then they request a record from DNS server by sending a 60B UDP DNS query packet. As a query response the victim might be sent 122B of A record, 4000B of TXT record and 222B of SOA record resulting in 73 times larger answer than request. Theoretically if the attackers generate 140Mbps the victim is flooded by 10Gbps.[17]

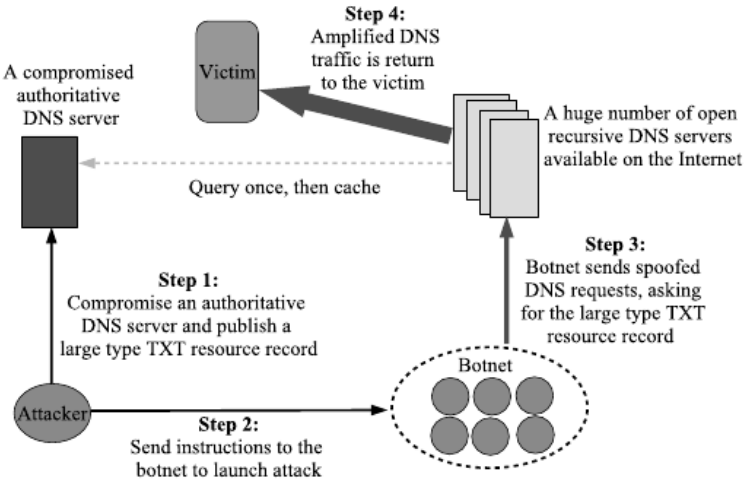


Figure 3.1: An ilustration of distributed reflected denial of service attack by[17]

3.3.3 HTTP flood

The most popular service brought by internet is no doubt HTTP used transfer webpages from one PC to another. It runs on port 80, which is open on almost every PC on the internet. Hence it is a possible target of the attackers, which use their zombies to generate HTTP requests. To amplify the attack the zombies might be instructed to download a large file from the victim, which then has to read it from the hard-drive, store it in it's memory a fit it into packets. This results in consumption of victim's resources. This type of attack appears very genuine and it is nearly impossible to tell which connection is legitimate a which is not.[17]

3.3.4 Authentication flood

Many servers offer services like email accounts, where it is necessary to use authentication to tell if the person trying to log in is allowed to. Authentication is also an expensive operation for the server often has to access a database placed on a hard-drive to verify submitted data.

3.3.5 Search attack

Search attack is a classic application level attack. If the victim offers searching either through contents of it's website or the entire internet the attackers may instruct their zombies to issue searching. The search operation is often very resource consuming and

occupies CPU, hard-drives and memory. The servers are not capable of handling too many search operation simultaneously and therefore tend to overload.

3.3.6 SIP flood

Session Initiation Protocol was developed due to the need of connecting VoIP calls. As the popularity grows it is becoming an attractive target for attackers. In order to understand the attack couple details need to be mentioned about the way a call is initiated. First the caller sends an invite packet to his or her SIP proxy server. Caller's SIP proxy looks up receiver's SIP proxy server and forwards the invite packet to it. Finally the receiver's SIP proxy server forwards the packet to the receiver and his or her IP phone will ring. When executing SIP flood the attackers overwhelm the SIP proxy server with invite packets with spoofed source IP addresses. The servers become unable to server the legitimate callers as their resources are depleted by serving the attackers fake calls. [17]

3.3.7 Buffer overflow

Buffer overflow is one of the easiest to execute and most common attacks. Even 12 year old children are able to commence such attack. All they need is a piece of code, which is to be found on underground websites along with step-by-step manual. Buffer overflow attacks are possible due to irresponsible programmers who do not check size of input from user. The most common case is when asking a user for authentication. The programmer uses a buffer of let's say 50 or 100 characters thinking that no one uses longer username than 10 characters and that there is plenty of room for couple extra characters for paranoid users. As he or she copies the input to the buffer, the input overwrites the memory located right behind the buffer. This can in the least harmful case cause the application to collapse and thus deny service. However there are much more serious issues. The attacker can if clever enough overwrite return address of the calling function with an address of his or her own code. This way is the attacker able to install backdoor programs, get users' data or even format the hard-drives.[1]

3.3.8 U2R

U2R is an abbreviation for user to root attack. The attacker logs in as a regular user and uses an exploit to gain root's rights. Then nothing stands between the attacker and his goal, which might be denying service, getting sensitive data or destroying them.

3.3.9 R2L

R2L(remote to local) is an attack when the attacker uses an exploit to fool the system and make it think that the attacker is locally present on the machine even though he or she is accessing the remotely.

R2L, U2R and buffer overflow are attacks detectable by searching for patterns inside the packets' payloads. They were mentioned for completeness of the list of attacks. Their detection is outside the scope of this thesis. That is mainly due to the fact that the detection algorithm would have to look inside the payload for these attacks seem perfectly normal by statistical evaluation of NetFlow data as they consist of few packets.

3.4 DDoS today

DDoS is experiencing a change. There is less simple flooding attacks and more sophisticated attacks such as service and application level attacks. The ISPs do not see DDoS as their biggest problem as the last security survey shows. Due to better communication, collaboration and previous infrastructure investments the ISPs are now more capable of protecting services of their customers. We have learned that DNS amplification is still very popular. And as the figure 3.2 below shows, the largest attacks generate over 40Gbps of data. That is 100times more than in 2001.[13]

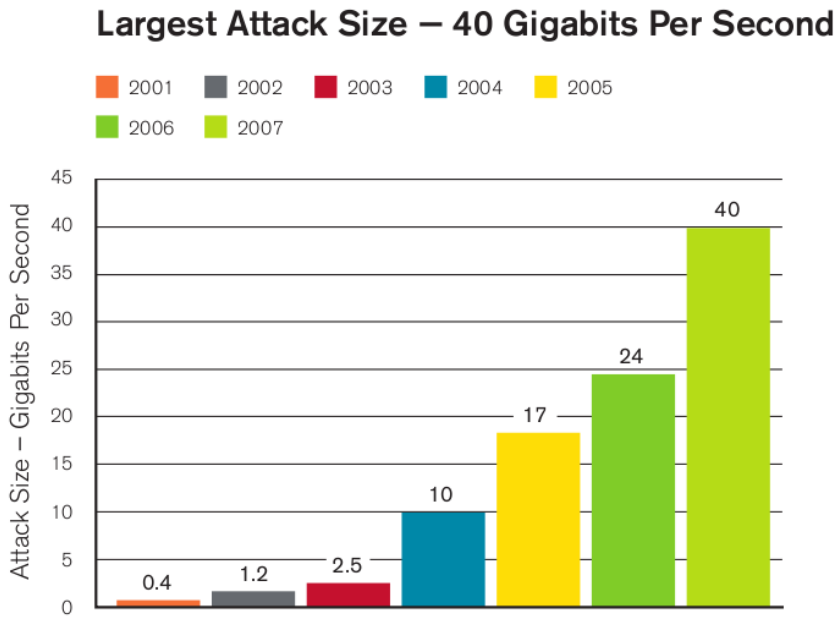


Figure 3.2: Maximum data generated by DDoS over past several years according to Arbor Networks Inc.

In 2007 was DDoS attack used for the time against a country namely Estonia. The Estonian Government had decided to move a Soviet monument from the center of Tallinn to outskirts of city and thus angered the Russian minority and all of Russia. After couple days of riots massive DDoS attacks were launched against Estonian government websites. 10 of those attacks exceeded 90Mbps and lasted over 10 hours. Some authorities claim Russia was involved in this cyber-warfare. [15]

DDoS has become a part of modern warfare as seen in the Russia-Georgia conflict in 2008. In this warfare the cyber attacks and real shooting coexisted for the first time. Several Georgian government servers were targeted as well as president Mikheil Saakashvili's website. Bill Woodcock said DDoS is likely to become a part of modern warfare for its inexpensive, anonymous and simple nature. [12]

It is impossible to take actions that would stop DDoS once and for all, but it is possible to make it so difficult for the attackers that only the best prevail. And Hence significantly reduce the number of DDoS attacks.[14]

Chapter 4

Network attack detection

As you may have noticed by reading the previous chapter, there are many types of network attacks. So the goal in network attack detection is to create an IDS, capable of detecting all or at least most of these attacks. There are many approaches with great results, but there is still room for improvements. The detection approaches fall into two basic categories - anomaly-based and signature-based.

4.1 Signature-based detection

Signature-based detection was the first type of IDS widely deployed. It uses signatures(patterns) of network attacks to detect them. It basically searches for strings in the packets' payloads that are known to be malicious. These signatures have to be manually added to the IDS's signature database in order for the IDS to recognize attacks. Once a signature of an attack is entered this attack will never again occur unnoticed. That is a great quality of an IDS. There is a great but coming. Signature based IDSs are unable to recognize any attack for which an operator has not entered a signature. So if a new type of an attack appears on the network no one may notice. Moreover if an attack, for which a signature is already in the database, differs enough, it will not be detected. This poses a serious problem as the attackers may, to some degree, use variations of the attacks to avoid detection. Even bigger problem pose worms with self-alternating behavior.

Entering signatures is a time consuming activity for many network experts and thus a very expensive task. One might expect no false positives, but that is not the case here as the signatures are based on regular expressions. System requirements are used to be fairly low since the IDS looked through the payloads for pack of patterns. The attackers have figured out a way to make the signature-based IDS sweat when detecting. Namely by either encrypting the payloads or splitting them.

If protecting a system using just couple protocols, the signature database shrinks and the requirements lower. A typical signature-based IDS is SNORT.[2]

4.2 Anomaly-based detection

The market is demanding an IDS capable of detecting new attacks while having a low number of false positives. This brought attention to data mining techniques as means of intrusion detection. Anomaly detection recognizes new attacks, but suffers higher false positive rates. Lowering the number of false positives has been a priority for some time

now.[**?**] Anomaly-based detection tools are further divided into two categories. Supervised and unsupervised anomaly detection.

4.2.1 Supervised anomaly detection methods

First needs labeled data to train on. During training it is creating a profile for each class. The classes might be: normal, DoS, Probes, R2L and so on. A set of test data is used after the classifications have been created to determine how well the algorithm performs in terms of false and true positives. The greatest problem is obtaining a dataset to train on. When it is desired to have a profile for each type of attack the dataset for training must contain instances of each of those attacks. Moreover it has to carry a label saying what kind of an attack is it. Even bigger issue than labeling might be finding instances of each attack as they might not occur on the particular network segment. It is very difficult to obtain data from a network where you are not an administrator, because most admins will not let you use their network traffic records. They simply fear a misuse. The IDS designer often has no other option than to simulate the attack himself.

Classifications of various attacks are not necessary. It is sufficient using just two classes: anomalous and normal. When having just the two classes the dataset needs to be intrusion free. If not, future occurrences of the attack buried within the training data might not be detected. It is almost impossible to find a clean dataset containing no attacks to train on.

So you either have to get a group of network experts to label a dataset or to go through it to check if it is clean. Another problem of supervised anomaly detection methods is inability to detect highly altered variations of attacks. The number of occurrences of words such as very difficult, almost impossible in the past two paragraphs suggests that this is not the best way to go. For completeness I feel obligated to mention algorithms used in this area, which are decision trees, neural networks and pseudo-bayes estimators.[**?**]

4.2.2 Unsupervised anomaly detection methods

Unsupervised anomaly detection methods do not need as opposed to supervised labeled data to train on. The training dataset may even contain attacks. The attacks present in the training dataser will be detected later on as long as normal data represent the majority of the dataset. Another necessity for the methods successful detection is statistical deviation of the attacks otherwise they might be considered legitimate. A great property of these algorithms is the ability to detect unknown and previously unseen attacks as long as they differ from the norm. What has been a problem for a very long time is number of false positives. These techniques have in the past several years received much attention focused on reducing FP rate. A concrete example of unsupervised anomaly detection algorithm is TCM-KNN, which is described in the next chapter in detail. [7]

Chapter 5

Design of the anomaly detection module

I decided to implement TCMKNN algorithm as described in [11] and created an IDS NetWarden. The algorithm is based on detecting outliers in a N-dimensional space. Each flow is represented by point in this space. The number of dimensions is equal to the number of features - attributes - used for the flow's description. Flow's position in this space is determined by it's attributes. For better understanding see figure 5.1. It is a simplified snapshot of the N-dimensional space where N is 3. Each dimension corresponds to one feature - f1,f2 and f3. Flows with similar characteristics are situated close to each other and form clusters. These clusters represent normal behavior, but it does not necessarily have to be just normal behavior. That depends on the features used as certain features expose only certain types of attacks. Remote to local attack for example is for NetWarden undetectable and by the used features seems as perfectly normal traffic. This particular attack might be and most likely will be (if an instance of it occurs on the protected network segment) part of a cluster of legitimate traffic. If a feature exposing an attack is present it moves the flow representing the attack away from the cluster. A vertical port scan(one machine different ports) for example will have much larger value of feature stating number connections made from one source IP address to one destination IP address but always to a unique destination port. Presence of such characteristic will ensure detection of such behavior, which would then appear as an outlier - o1 or o2. c1 and c2. o1, o2 and the rest of points outside of clusters are called outliers and represent anomalous behavior.

To determine if a flow is an anomaly several computations are made. In order to understand the following formulas couple declarations must be done. F is a set of flows including flow representing cetroids(explained later on). T is a set of features used for flow description. D_{fn} is the nth shortest distance to a neighboring flow.

First the distances to the K closest neighbors are determined using Euclidean distance:

$$D_{f_1,f_2} = \sqrt{\sum_{j=1}^{|T|}(f_1[j] - f_2[j])^2} \tag{5.1}$$

where is number of features used, f_1 and f_2 flows(or a flow and a centroid) of which the distance is desired and j a feature. Before the distance calculation feature normalization must be done. That prevents domination of one feature over the others. Number of

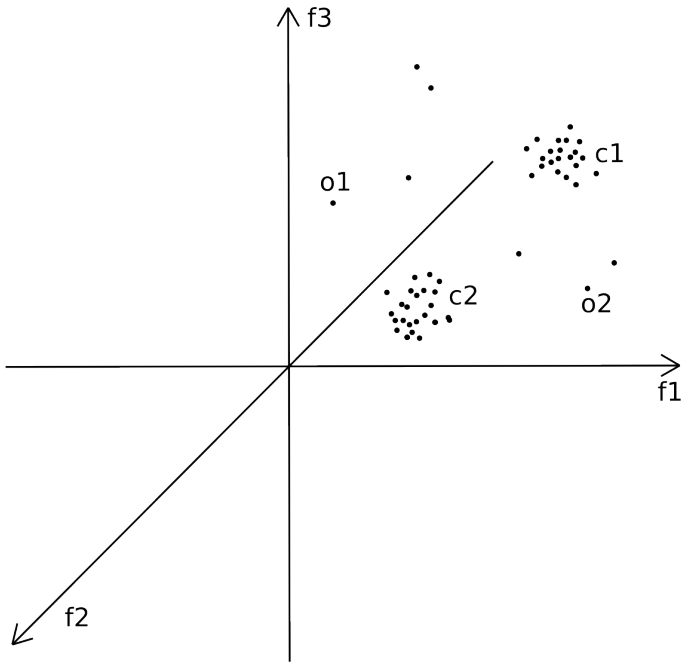


Figure 5.1: Flows represented as points in a 3 dimensional space

transferred bytes for example usually differs by a lot. A user might send a request of couple hundred bytes or upload a picture with couple hundred kilobytes of size. Protocol on the other hand is usually TCP(0) or UDP(1), which means they are very close. Without normalization it would be pointless to even include such feature as protocol for it would never make a difference. To normalize features we need to compute mean value and standard deviation value for each feature.

$$mean[i] = \frac{1}{|F|} \sum_{i=1}^{|F|} f_i[j] \tag{5.2}$$

$$standard[j] = \sqrt{\frac{1}{|F|-1} \sum_{i=1}^{|F|} (f_i[j] - mean[j])^2} \tag{5.3}$$

Normalized features would be then calculated by the following formula:

$$f_{normalized}[j] = \frac{f[j] - mean[j]}{standard[j]} \tag{5.4}$$

Once all the flows know how far their K nearest neighbors are it is time to evaluate which are outliers(anomalies possibly attacks) and which are(or seem) legitimate. The strangeness value is then calculated according to this equation:

$$\alpha_f = \sum_{n=1}^{K} D_{fn} \tag{5.5}$$

where D_{fj} stands for nth shortest distance from flow f. It is logical to use such measure because outliers will have larger distances to their neighbors resulting in higher strangeness value. Based on strangeness value a pvalue is calculated according to the following formula

$$p_{f_{new}}(\alpha_{f_{new}}) = \frac{|\{f : \alpha_f > \alpha_{f_{new}}\}|}{|F| + 1} \tag{5.6}$$

where $p_{f_{new}}$ is a flow pvalue of which we are calculating, $\alpha_{f_{new}}$ is it's alpha value, α_f an alpha value of all present flows. The numerator basically states a number of stranger flows(flows with larger strangeness value). n is the number of all flows.

5.1 Learning process

First step is creating a model of the normal network behavior, which is common for most ID systems. Building a model means in this case reading NetFlow data containing input files. NetWarden handles either files produced by the nfcapd or already in text format produced by nfdump. The flow notation is demanded in the following format: flow initiation date and time, duration, protocol, source IP address:port, destination IP address:port, number of packets, number of bytes and finally flags. When feeding NetWarden text files processed by nfdump it is imperative to have nfdump sort the flow records chronologically.

The next step is deriving advanced features for each flow. Concrete features are in listed in 5.3.1 and in 5.3.2 and 6.2 and will be described later in great detail. In order to derive the time-based and connection-based features NetWarden needs to have a context loaded. Context would be a certain number of input files preceding the currently processed one. For time-based features a context of size 1 is sufficient for it looks just 5 seconds back. The connection-based features though need much longer context. The connection-based features used expose slow network scans. The context size directly affects the slow scan detection sensitivity. The longer the context the slower scans are detected. Note that size of the context window depends on the machine running NetWarden. If equiped enough in terms of RAM the size of the context window might be quite large. Once the features are derived a flow record is saved to become a part of the model.

As the model grows, more and more points appear in the N-dimensional space forming clusters representing(hopefully just) normal behaviour on the controlled network segment. On network segments with high bandwidth links such model can reach an enormous size. Dropping records is a possible solution, which might cause undesirable rise of the number of false positives. That would be an unfortunate outcome. Far better solution is substituting set of flows close to each other - cluster - for one with the same properties - centroid. The logic behind it is very simple as illustrated in figure 5.2. The centroid is defined by the

same set of attributes as a single flow. The only difference is in weight. Weight tells us how many points the centroid represents. A flow is added to a centroid if the distance between them is smaller than a specified clusterization threshold. This threshold is set upon NetWarden's execution. Obviously the larger the threshold the lower the memory requirements, but also the detection qualities. Larger clusterization threshold also means less time need to process input data as the model contains less objects(centroids or single flows if they are not close enough to any other) to compute distance to. To get an idea how much influence the threshold has on TP, FP memory requirements and processing speed see chapter 6. Centroids' features are updated with addition of each new flow according to equation 5.7.[16]

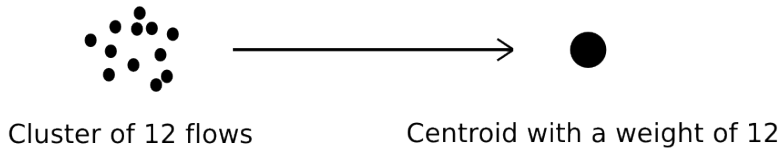


Figure 5.2: Substitution of a cluster of flows for a single centroid with the same properties

$$c_i[j] = \frac{c_i[j] * c_i[weight] + f_{new}[j]*}{c_i[weight] + 1} \tag{5.7}$$

All flows fed to NetWarden during the training period are included in the model either in it's original form or as a part of a centroid. It is imperative that a network administrator examines the data fed to NetWarden for intrusions. If there are just few instances of each attack it should not affect detection of their instances in the future. How many is a few depends on the number of K-neighbors used. The more neighbors used the more instances of the attack is NetWarden capable of having in it's model.

5.2 Detection

As soon as the training period expires NetWarden has an idea of what normal traffic on the particular network segment looks like. It is considered best practice to let NetWarden learn for a week. Week is a very significant time period when modeling behavior not only on the network. It usually contains most traffic seen on the network and thus resulting in less false positives. That is mainly due to the nature of the detection algorithm. It detects anomalies and when previously unseen traffic is encountered it will cause an alarm even if it is perfectly legitimate traffic.

Newly loaded input files after the detection has started undergo the mostly the same procedures as during training. The advanced features for each flow are derived and normalized and distances of all neighbors are acquired, but are saved only K of the closest ones. This step is very computationally expensive. Therefore random sampling has been deployed. When executing NetWarden it is possible to specify the degree of sampling. 100 is a default value, which means that when reading the input file 1-100 flows are omitted.

This is necessary for files containing tens of millions of flows per hour and more if you want to keep the detection real-time.

The next step is strangeness value computation as described earlier. When all flows and centroids have their alpha value assigned pvalue might be determined. Each newly loaded flow's pvalue is compared to the anomaly threshold and if it is larger it means we have stumbled across an anomaly. Anomalies are reported to the network administrator through a simple CLI. If the anomaly is merely unseen legitimate traffic, the administrator can add an exception, preventing NetWarden from reporting it again. Non anomalous flows are added to the model.

The CLI allows the network operator to list anomalies, make exceptions out of the anomalies, list exceptions and also delete exceptions when no longer needed or when they become a potential security risk.

NetWarden has been designed with expandability in mind. While it is for now using just one detection method I would like to add more in my future work to improve it's detection qualities. Each method's pvalue depending on it's accuracy will have corresponding weight. If the resulting pvalue computed as a sum of pvalues supplied by each method multiplied by the method's judgment weight is higher then a specified threshold it will be reported as an anomaly. This idea was inspired by authors of[19].

Each flow or cluster keeps as many distances as is the number of K-neighbors used. These are distances to the closest points. It is apparent that if a flow is inside a cluster it's neighbors are close and the distances to them are small. This fact is used when determining how strange the flow is and each flow is assigned a strangeness value.

5.3 Features

Together with choosing a suitable algorithm a feature selection is the most important task when building an effective anomaly-based IDS. Many features have been proposed by people with extensive knowledge of the intrusion detection system architecture so one can either choose from theirs or design his or her own. First step in feature selection is analyzing behavior of current attacks and of course attackers. Since the attacker usually scans the victim's network before attacking it is good to have an ID system capable of detecting these probes. Scan detection tells you that something is coming up and gives you time to prepare. Network attacks can be mostly divided into several groups, where each group is detected by a different set of features. The largest group is DoS, which is detectable by number of connections to a single host(that is any source IP address but one destination IP address). Next feature exposing DoS would percentage of connections to the same host to the same service. And let's not forget number of bytes transferred to the host.

It is important to have several features for each groups as it is very common for legitimate traffic to have certain behavioral similarities with intrusions. A web server on a network segment is a classic demonstration of the above. While for a web server numerous connections to the same host and same service are quite normal, flows representing legitimate traffic from users simply browsing your company's webpages might be mistaken for DoS. That is if only a feature stating number of connections to the same host and service were relied on for DoS exposure. The bottom line is that the more features the better chance of detecting particular group of attacks and the less chance of legitimate traffic misclassification.

Although having a lot of various features is generally(if properly designed) improving the ID systems' detection and false alarm rates, it is often rather computationally expen-

sive. The reason is simple, number of features reflects number of dimensions. The more dimensions used the longer it takes to compute the distance between two point - flows. The ID system with too many features might be unable to operate in real time, which is more and more difficult as the bandwidth increases. Gigabit connection to the internet become more and more common these days.

5.3.1 Basic flow characteristics

The basic flow characteristics play a key role in detecting anomalies. They not only serve to point out abnormal behavior on the network, but are also necessary for mining the advanced features described later.

- **duration**
- **protocol**
- **source IP address**
- **source port**
- **destination IP address**
- **destination port**
- **number of bytes transfered**
- **flags**

5.3.2 Time-based features

When deriving these features a time window of x seconds is used. That tells us how intense certain activities are. These features were designed to capture traffic that would seem perfectly normal when using only the basic features to describe behavior on the network.Those include time-based features:

con_src_t will alert us if a target of DoS or DDoS that is being executed is located on our network segment. Regular users are able to make only limited amount of connections in 5 seconds. Problem may arise when a user is using programs that make numerous connections in very little time with no harmful intentions whatsoever. Misclassification may occur.

con_dst_t is a feature saying how many connections were made to the same destination IP address as the currently examined flow. By including this feature NetWarden is able to detect DoS and DDoS attacks either using a large number of zombies or few machines using xxpoof-like programs. You might argue that the previous feature would do the trick. There is unfortunately a very easy way to fool the IDS using

con_src_t without **con_dst_t**. When using xxpoof-like program the attacker might for each connection use randomly generated IP address and thus effectively executed an attack unnoticed by the IDS. **con_src_t** would stay low for each IP address used. Together though it could be a devastating attack. Attackers with large enough botnets might use slower sending rate, but using thousands of zombies

byt_src_dst_t is a very straight forward feature showing how much traffic flows between the source and the destination. Having this feature enabled might detect DoS on network

feature	description
con_src_t	number of connections from the same source IP address as currently examined flow
con_dst_t	number of connections to the same destination IP address
byt_src_dst_t	number of bytes transferred from source IP address to destination IP address
byt_dst_src_t	number of bytes transferred in opposite direction
con_src_serv_t	number of connections to the same service(destination port)
con_src_dst_diffServ_t	number of connections to the same destination IP address but different service

Table 5.1: Time-based features and their description

segments with webservers, DNS and other services with generally low number of bytes per connection. Network segments accommodating mirrors for various software(FTP servers) would have flows with large amount of bytes transferred in their model of normal behavior. Hence brute force DoS would not be different from legitimate traffic according to this attribute.

byt_dst_src_t is much like the previous feature except for the direction, in which the bytes are sent. It has been included to alert us if for example a DRDoS was being executed using a DNS server on our network segment. The model will have port 53 asosiated with low number of bytes as answer for a name query and thus when the attacker asks for a 4000B TXT record an anomaly will be reported.

con_src_serv_t states number of unique destination IP addresses connected to from one source IP address using the same service. This feature is very likely to detect worms spreading thanks to an exploit in a certain service. As the worm spreads it searches for vulnerable machines by examining if the particular service is unprotected. Horizontal network scan is also likely to be exposed unless the attacker is patient and scans slowly(this feature has just five seconds of context).

con_src_dst_diffServ_t was designed to detect vertical scan. So it holds number of connections from the same source IP to the same destination IP, but to a unique service(destination port number).

And finally the connection-based features. These features have been proposed mainly for slow scan detection. Let's say the attacker scans one port per minute. As the time-based feature looks back just five seconds, they will detect nothing. The connection-based features on the other hand look back hundred connections from the same source. I have used the following:

5.3.3 Connection-based features

con_src_serv_c states number of unique destination IP addresses connected from this source IP address using the same service. It exposes horizontal scans.

feature	description
con_src_serv_c	number of connections to the same service(destination port)
con_src_dst_diffServ_c	number of connections to the same destination IP address but different service

Table 5.2: Connection-based features with their description

con_src_dst_diffServ_c is used to point out vertical scans by stating the number of unique destination ports that host with this source IP address connected to the same destination IP address.

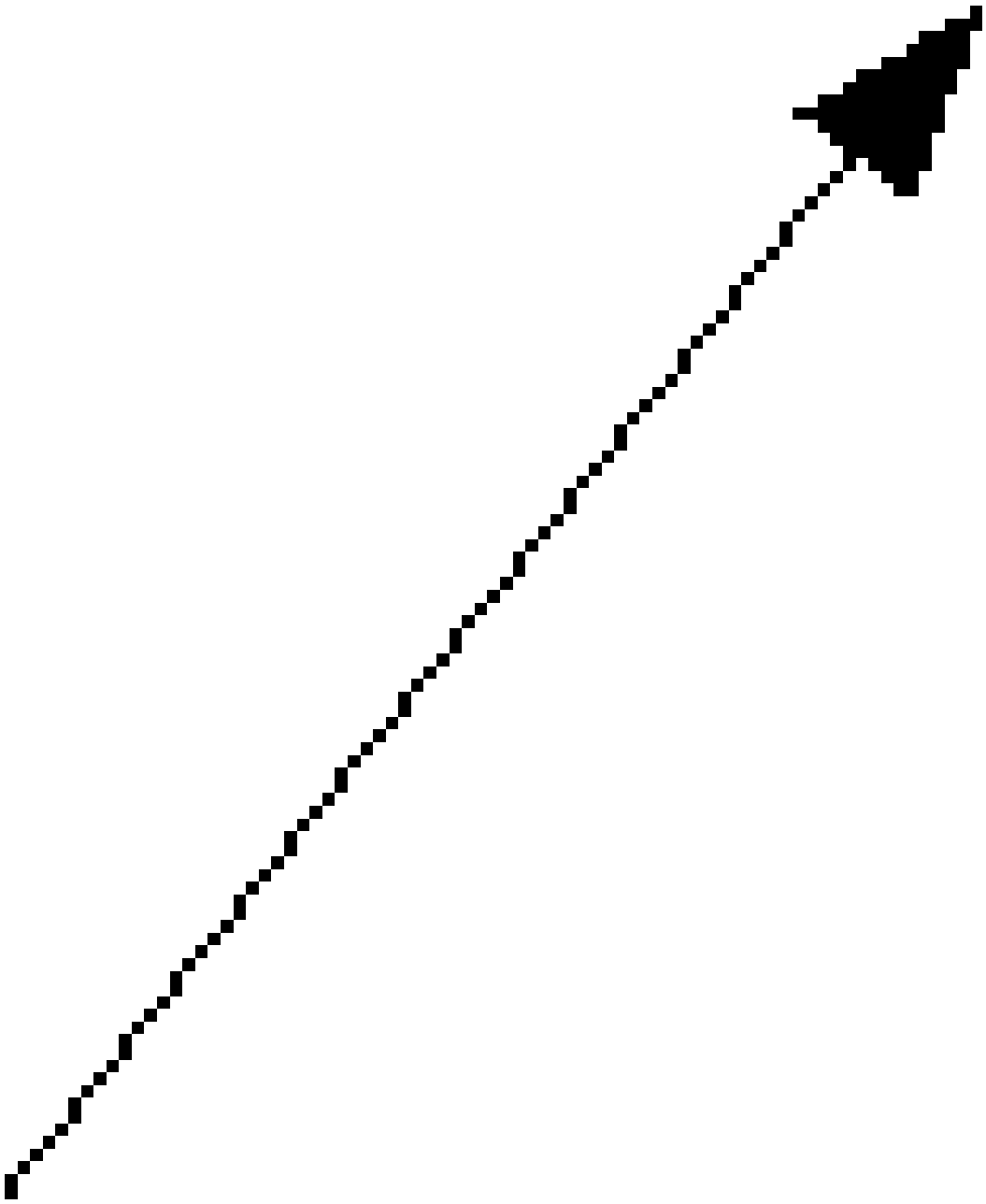


Figure 5.3: NetWarden's diagram

Chapter 6

Experimental results

Experiments are an very important part of this thesis and successful IDS creation. In order establish and maximize efficiency of my program it is crucial to have it analyze data from real network preferably the one it is designed for. There are several parameters of NetWarden that influence it's performance quite a bit. Settings such as clusterization threshold saying how close In order to achieve the best results I have first decided to run tests on KDD Cup 99 dataset.

6.1 KDD Cup 99 dataset

The KDD Cup is a competition in data mining and in year 1999 the goal was to create an IDS based on data mining. It has been used plenty of times since as a benchmark for ID systems. Though it has it's problems The dataset was created to represent traffic on a military network over seven weeks. It contains 24 different types of attacks in four categories. Those are: DoS, Probes, R2L and U2R. The last two are for this program undetectable due to their nature as described above and hence the results are worse than achieved in [11]. The KDD99 dataset was originally in tcpdump format and was preprocessed for the competition by Bro IDS. As a result each connection is described by 41 features. I have used only the ones obtainable from the IP header, which are identical with features derived from NetFlow data. The features used for the evaluation are the following:

Experiments on this dataset were done for the following reasons. First it is quite simple to check the results and thus is priceless in determining what the best values for the thresholds are. And second it is good to see how NetWarden does detection-wise compared to other ID systems evaluated on this dataset.

I have used the 10% datasets. Namely the kddcup.data_10_percent.txt for training and corrected for testing. It is important to note that only normal connections were built into the model so the performance was better than on real NetFlow data, which are mostly at least a little noisy. The authors in [11] say the algorithmn handles up 1.5% of noise in the dataset. I would go farther and said that the acceptance of noise depends on the number of neighbors used. Note that the slightly larger k results in much larger memory requirements.

I have experimented with the following attributes of NetWarden. First was clusterization threshold, which as it turned out has a great impact on NetWarden's performance in terms of processing speed and memory requirements. It has somewhat worsened the detection qualities as you can see in the chart below. The ROC expresses dependence of TP and FP on a variable. The Y-axis represents TP and the X-axis FP. The best result is reached

feature	description
duration	Duration of the connection.
protocol	Connection protocol - TCP,UDP,ICMP etc.
source bytes	Number of byter transferred from source to destination
destination bytes	Number of byter transferred from destination to source
count	Number of connections to the same host as the current connection in the past two seconds
srv count	Number of connections to the same service as the current connection in the past two seconds
same srv rate	% of connections to the same service
diff srv rate	% of connections to different services
srv diff host rate	% of connections to different hosts
dst host count	Number of connections having the same destination host
dst host srv	count of connections having the same destination host and using the same service
dst host same srv rate	% of connections having the same destination host and using the same service
dst host diff srv rate	% of different services on the current host
dst host same src port rate	% of connections to the current host having the same src port
dst host srv diff host rate	% of connections to the same service coming from different hosts

Table 6.1: Features used during KDD Cup 99 evaluation. Description from [8]

when the curve in on point or another hits the top left corner. That is 100% TP and 0% FP.

Each curve in the chart above represents a set of experiments done with clusterization thresholds 0.1,0.2 and $0,3$ respectively. Each curve had a constant value of anomaly threshold, which are listed in the chart's legend. As you can see TP and FP rate worsen with larger clusterization threshold as expected. The larger the threshold the greater chance of an anomalous flow to be within this range. The system requirements and processing time were also affected by the change in clusterization. See the table below for more details.

Different values of anomaly threshold had very little little impact on the detection qualities. It is due to the fact that most of the connections representing attacks were extremely anomalous and thus detectable even with anomaly treshold set to 0.99. When analyzing traffic on real network, setting the higher or lower will result in less or more alarms respectively. The number of neighbors used was kept constant with a value of 1000

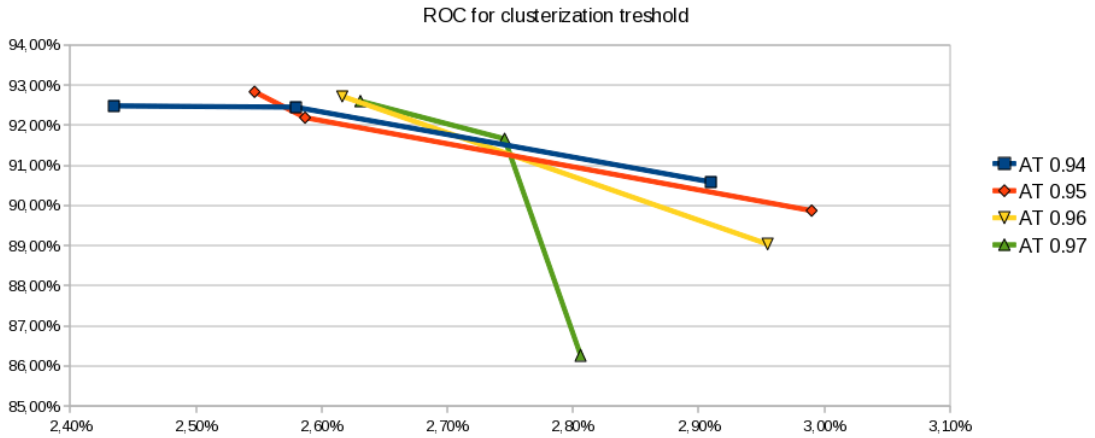


Figure 6.1: ROCs of clusterization threshold

clusterization threshold	processing time	memory usage
0.1	37 minutes	2.6 GB
0.2	27 minutes	2.3 GB
0.3	11 minutes	1.4 GB

Table 6.2: Memory requirements a processing time for different values of clusterization threshold

as in [11].

Next set of experiments were dedicated to see what effects will changing K have. Constant values of anomaly and clusterization treshold were used - $AT = 0.95$, $CT = 0.2$. The following chart displays ROC depending on K.

As we can see from the chart the best results were achieved with $K = 1000$. TP was higher for $K = 1200$, but this slight improvement in TP worsened FP, which more important. So the answer is that 1000 is an optimal number of neigbors to use to get the best results.

6.2 Real NetFlow data

I have obtained NetFlow records from a private network for NetWarden's further evaluation and experiments. I must note that getting a dataset from a suitable subject such as an ISP is extremely difficult due to privacy and security issues. Acquiring number of false positives and true positives is not as easy and clear with real data as it is with the KDD 99 dataset, where all the flows are labeled. Determining It means going through the anomaly log and checking every record. Most of the reported flows represented interesting behavior. I would not go as far as to calling it an attack, but it was definitely unusual and thus interesting to the network administrator. It was a false positive nonetheless, but it's detection tested the very nature of the used algorithm and by that logic it did very well. It detected an anomaly.

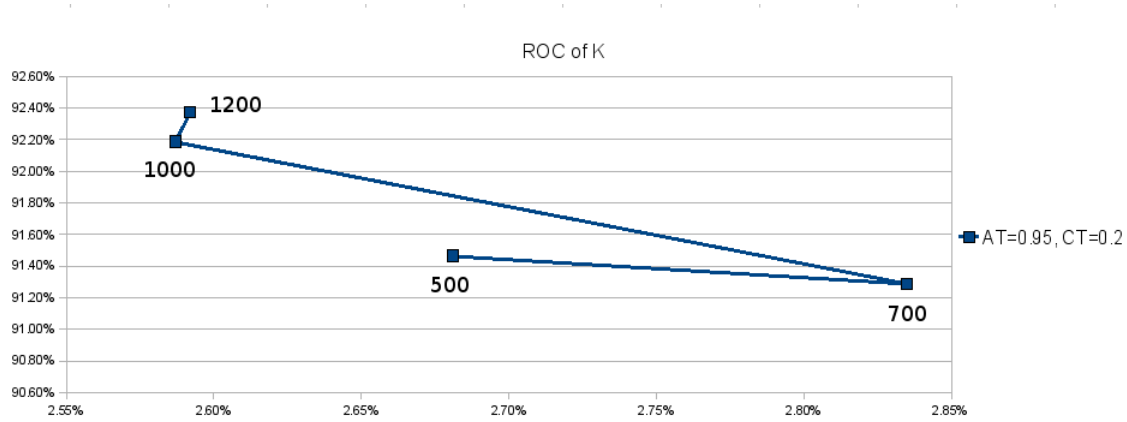


Figure 6.2: An ROC of K

Though not a large dataset it served it's purpose well. It has been gathered over a 34 hours and contains altogether 439619 flows. This data come from a well secured network so it is very unlikely for an attack to appear in it. The number of anomalies for that reason will be equal to number of FP. Training period was one day, which explains fairly large FP rates. I have decided to see what features make particular flows look like an anomaly and how much they generaly affect the output. In the first run all features were enabled and an anomaly log was saved. With all features on 132 anomalies were reported. That is a very low number due to the anomaly threshold being set to 0.98. The table below shows for each omitted feature how many anomalies match the anomalies reported with all features enabled.

Feature omitted	**anomalies reported**	**common**
con_src_t	350	0
con_dst_t	324	0
byt_src_dst_t	177	0
byt_dst_src_t	139	1
con_src_serv_t	344	24
con_src_dst_diffServ_t	154	0
con_src_serv_c	478	0
con_src_dst_diffServ_c	198	18

Table 6.3: Memory requirements a processing time for different values of clusterization threshold

Now let's analyze the table contents. It is safe to say that if number of common anomalies is high the feature does not have too much of an impact on NetWarden's judgement. The features **con_src_dst_diffServ_t** and **con_src_dst_diffServ_c** have made NetWarden confuse regular https traffic for a vertical scan. Internet browser opened numerous connec-

tions to the server and when the server answered it appeared as a probe. When any one of these were turned off most of these flows looked normal.

con_dst_t and **con_src_t** make connections to webservers and their responses appear normal. When inactive the above mentioned presumed vertical scan was reported much more. It turned out that the original assumption of these features helping detect DoS was incorrect. But they have proven useful in another way.

Omitting **byt_src_dst_t** and **byt_src_dst_t** resulted in a slight change of anomalies reported but in this case did not play any key role. These features would need to be tested on a larger dataset to draw any relevant conclusions.

Both **con_src_serv_c** and **con_src_serv_t** had an unneglectable impact of the number of alarms. That is due to the fact that they have while on contributed with a majorly to detection of some highly anomalous flows. Those flows did indeed appear as a worm or a horizontal scan although I have been assured the data is intrusion free. The problem is that when off the anomalous flows that appear by every other feature normal are brought into a cluster and thus undetected.

As it turns out designing an IDS is not only about features exposing attacks, but also features that ensure that connections that are normal are classified so. Careful feature design is extremely important to correct anomaly detection.

I have estimated throughput based on the experiments. With no sampling the NetWarden handles 5000 flows per minute, which is a very small throughput and it is an subject to future attention. Possible solution is rewriting distance computing in assembly.

All tests were run on Core 2 Duo T8400 - 2.4 GHz, 3GB RAM with Ubuntu Linux 9.04.

Chapter 7

Conclusion

NetWarden has proven fairly accurate. The results of experiments of the KDD 99 dataset were satisfactory. FP and TP rates are comparable to other projects. NetWarden is based on an unsupervised anomaly detection method, which makes it perform well even on slightly noisy dataset. Tests done on the real NetFlow data resulted in undesirebly large number of false positives. That is not always a bad thing. Eventhough not attacks reported flows represented behaviour interesting to the network administrator. Overall it detects well with limitaions in throughput.

The greatest downside as with other IDSs remains high number of false positives. Lowering it is a priority for future work. NetWarden has been designed to be expandable, which is also the first line on the TODO list. Adding another module will improve NetWarden detection qualities resulting in lower number of FP. Another goal is adding a GUI with visualisations of the traffic examined and aggregation. Most importantly system requirements must be lowered and throughput enlarged. Last but not least is adding support for IPv6.

The intrusion detection is a challenging field. It is a very active area of research. Unsupervised anomaly-based detection has received great attention lately bringing many improvements. I have become aware of is done behind the scenes to keep my PC safe and want to become part of it.

Bibliography

[1] Buffer overflow exploits: The why and how.
http://www.mcafee.com/us/local_content/white_papers/wp_ricochetbriefbuffer.pdf/.

[2] Ids: Signature versus anomaly detection.
http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1092691,00.html.

[3] Netflow. http://en.wikipedia.org/wiki/Netflow.

[4] Netflow. http://cs.wikipedia.org/wiki/Netflow.

[5] scapy. http://www.secdev.org/projects/scapy/.

[6] Abbas.
Sleeper virus: 8 million windows machines infected with downadup or conflicker worm!
http://tnerd.com/2009/01/17/remove-sleeper-downadup-conflicker-wrom-from-microsoft-

[7] Levent Ertöz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Vipin Kumar, Jaideep Srivastava, and Paul Dokas. Next generation data mining, 2004.

[8] H. Güneş Kayacik, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. http://www.lib.unb.ca/Texts/PST/2005/pdf/kayacik.pdf.

[9] Brian Krebs. 'sasser' worm tip of the pc bug invasion.
http://www.securityfocus.com/news/8573.

[10] Robert Lemos. Worm worries grow with release of windows hacks.
http://news.zdnet.com/2100-1009_22-135774.html?tag=nl.

[11] Yang Li, Binxing Fang, Li Guo, and You Chen. Network anomaly detection based on tcm-knn algorithm. http://portal.acm.org/citation.cfm?id=1229292.

[12] JOHN MARKOFF. Before the gunfire, cyberattacks.
http://www.nytimes.com/2008/08/13/technology/13cyber.html?_r=1.

[13] Danny McPherson and Dr. Craig Labovitz. Worldwide infrastructure security report, volume iv, 2008.

[14] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, December 30, 2004. ISBN 0-13-147573-8.

[15] Jose Nazario. Estonian ddos attacks - a summary to date.
http://asert.arbornetworks.com/2007/05/estonian-ddos-attacks-a-summary-to-date/.

[16] VLADIMIR NIKULIN. Weighted threshold-based clustering for intrusion detection systems. http://www.worldscinet.com/ijcia/mkt/free/S1469026806001770.pdf.

[17] TAO PENG, CHRISTOPHER LECKIE, and KOTAGIRI RAMAMOHANARAO. Survey of network-based defense mechanisms countering the dos and ddos problems,acm comput. surv. 39, 1, article 3 (april 2007). http://doi.acm.org/10.1145/1216370.1216373.

[18] George Varghese Ramana Rao Kompella, Sumeet Singh. On scalable attack detection in the network. http://www.imconf.net/imc-2004/papers/p187-kompella.pdf.

[19] Martin Rehák, Dr.Michal Pěchounek, Pavel Čeleda, and Vojtech Krmíček. Camnep: An intrusion detection system for highspeed networks. http://www.nii.ac.jp/pi/n5/5_65.pdf.

[20] Erik Rodriguez. Howto - spoofed dos attacks. http://www.skullbox.net/spoofeddos.php.

Appendix A

CD contents

The CD contains NetWarden's source files including a makefile, a set of test data, README, NetWarden's documentation and a pdf of this thesis.

Appendix B

System requirements

NetWarden has been tested on Linux Distribution Ubuntu 9.04, but should run on most Unix OS. The source files are located on the CD enclosed with this thesis. For compilation use Makefile. Make sure you have g++ installed. NetWarden needs libboost-thread-dev to successfully compile. libboost-thread-dev ought to be in most linux distributions' repositories so you may use your favorite package manager to obtain it. It is important to install version 1.35 or newer. If you wish to work with data files directly from nfcapd you will need to install also nfdump tools, which again should be in the repositories.

Appendix C

Manual

NetWarden is controlled is through command line. It accepts following arguments:

-s <file>	37 Required argument with source file containing NetFlow. The file is expected to be in format '*.YYYYMMDDHHMM*' with every file containing 5 minutes of data.
-k <number>	Set number of neighbors. Default is 1000.
-c <decimal number>	Set clusterization threshold to <number>. Default value is 0.1
-a <decimal number>	Set anomaly threshold to <decimal number>. Default value is 0.95, which means that flows 'stranger' than 95% of the flows in the model shall be reported as anomalous.
-t <number>	Sets the training period length to <number> days.
-x	When feeding NetWarden with txt files.
-e <file>	NetWarden runs in evaluation mode over KDD Cup 99 datasets and as an input file expects data file for training and data in <file> will be then used as test data. The output

Example:
./detector -k 1000 -s /netflow/nfcapd.200809261315.txt -x -a 0.98 -t 1 -c 0.8 -f 10

C.1 UI

Shortly after executing NetWarden a simple user interface asks you what do you wish to do. This is main menu and the possible actions are:

- List anomalies
- List exceptions
- Quit

To get each menu you need to press key associated with it and return. The mode List Anomalies offers following actions:

- Next anomaly
- Add an exception
- Back

Adding an exception will no longer report anomalies that fall into the same cluster with the exception. Back takes you to the main menu from where you can go to the List exceptions menu. At List exceptions menu following options are available:

- Next exception
- Delete an exception
- Back

The entire UI is self-explanatory and very easy to get used to.